The UML as a Formal Modeling Notation

Andy Evans¹, Robert France², Kevin Lano³, and Bernhard Rumpe⁴

¹ Department of Computing, Bradford University, UK
² Department of Computer Science & Engineering, Florida Atlantic University, USA
³ Department of Computing, Imperial College, London, UK
⁴ Department of Computer Science, Munich University of Technology, Germany

puml@comp.brad.ac.uk

Abstract. The Unified Modeling Language (UML) is rapidly emerging as a de-facto standard for modelling OO systems. Given this role, it is imperative that the UML needs a well-defined, fully explored semantics. Such semantics is required in order to ensure that UML concepts are precisely stated and defined. In this paper we motivate an approach to formalizing UML in which formal specification techniques are used to gain insight into the semantics of UML notations and diagrams and describe a roadmap for this approach. The authors initiated the Precise UML (PUML) group in order to develop a precise semantic model for UML diagrams. The semantic model is to be used as the basis for a set of diagrammatical transformation rules, which enable formal deductions to be made about UML diagrams. A small example shows how these rules can be used to verify whether one class diagram is a valid deduction of another. Because these rules are presented at the diagrammatical level, it will be argued that UML can be successfully used as a formal modelling tool without the notational complexities that are commonly found in textual specification techniques.

1 Introduction

The popularity of object-oriented methods such as OMT [RBP+91] and the Fusion Method [CAB+94], stems primarily from their use of intuitively-appealing modelling constructs, rich structuring mechanisms, and ready availability of expertise in the form of training courses and books. Despite their strengths, the use of OO methods on nontrivial development projects can be problematic. A significant source of problems is the lack of semantics for the modelling notations used by these methods. A consequence of this is that understanding of models can be more apparent than real. In some cases, developers can waste considerable time resolving disputes over usage and interpretation of notations. While informal analysis, for example, requirements and design reviews, are possible, the lack of precise semantics for OO modelling makes it difficult to develop rigorous, tool-based validation and verification procedures.

The Unified Modeling Language (UML) [Gro97c] is a set of OO modelling notations that has been standardized by the Object Management Group (OMG).
It is difficult to dispute that the UML reflects some of the best modelling experiences and that it incorporates notations that have been proven useful in practice. Yet, the UML does not go far enough in addressing problems that relate to the lack of precision. The architects of the UML have stated that precision of syntax and semantics is a major goal. The UML semantics document (version 1.1) is claimed to provide a "complete semantics" that is expressed in a "precise way" using meta-models and a mixture of natural language and an adaptation of formal techniques that improves "precision while maintaining readability". The meta-models do capture a precise notion of the (abstract) syntax of the UML modelling techniques (this is what meta-models are typically used for), but they do little in the way of answering questions related to the interpretation of non-trivial UML structures. It does not help that the semantic meta-model is expressed in a subset of the notation that one is trying to interpret. The meta-models can serve as precise description of the notation and are therefore useful in implementing editors, and they can be used as a basis to define semantics, but they cannot serve as a precise description of the meaning of UML constructs.

The UML architects justify their limited use of formal techniques by claiming that "the state of the practice in formal specifications does not yet address some of the more difficult language issues that UML introduces". Our experiences with formalizing OO concepts indicate that this is not the case. While this may be true to some extent, we believe that much can be gained by using formal techniques to explore the semantics of UML. On the other hand, we do agree that current text-based formal techniques tend to produce models that are difficult to read and interpret, and, as a result, can hinder the understanding of UML concepts. This latter problem does not diminish the utility of formal techniques, rather, it obligates one to translate formal expressions of semantics to a form that is digestible by users of the UML notation.

In a previous paper, we discussed how experiences gained by formalizing OO concepts can significantly impact the development of a precise semantics for UML structures. We motivated an approach to formalizing UML concepts in which formal specification techniques are used primarily to gain insights to the semantics of UML notations. In this paper we present the roadmap we are using to formalize the UML, and describe the results of its application to the formalization of UML static models.

The primary objective of our work is to produce rigorous development techniques based on the UML. A first step is to make UML models amenable to rigorous analyses by providing a precise semantics for the models. This paves the way for the development of formal techniques supporting the rigorous development of systems through the systematic enhancement and transformation of OO models. In this paper we show how the formalized static model can be rigorously manipulated to prove properties about them and their relationships to other static models.

In Section 2, we present an overview of work on the formalization of OO modelling concepts and notations, and outline the PUML formalization approach.